Amendments to the Claims:

This listing of claims will replace all prior versions, and listings, of claims in the application:

Listing of Claims:

1. (Currently Amended) A method of evaluating a set of memory maps for a program having a plurality of functions, the method comprising:

(a) executing a first version of the program according to a first memory map to generate a program counter trace;

(b) converting the program counter trace into a program counter trace format defining a memory location in association with a function and an offset within the function using the first memory map;

(c) translating the program counter trace format into physical addresses using one of the set of memory maps to be evaluated, different from the first memory map;

(d) evaluating the number of likely cache misses ~~using a model of a direct-mapped cache~~ by passing the physical addresses for that one memory map to a model of a direct-mapped cache; and

repeating steps (c) and (d) for each of the memory maps in the set.

2. (Previously Presented) The method of claim 1, wherein step (c) is carried out by utilizing the base address of each function of said one of the memory maps to be evaluated with the offset given in the program count trace format.

3. (Previously Presented) The method of claim 1, wherein the direct-mapped cache model of step (d) emulates the operation of a cache such that would occur when a new version of the program linked according to said one memory map under evaluation is executed.

4. (Previously Presented) The method of claim 1, comprising the additional step of, subsequent to evaluating the first set of memory maps, generating a further set of memory maps for evaluation.

5. (Currently Amended) A method of operating a computer to evaluate a set of memory maps for a program comprising a plurality of functions, the method comprising:

loading a first version of the program into the computer and executing said first version to generate a program counter trace;

loading into the computer a memory map evaluation tool that carries out the steps of:

converting the program counter trace into a program counter trace format defining a memory location in association with a function and an offset within the function using the first memory map;

translating the program counter trace format into physical addresses using one of the set of memory maps to be evaluated, different from the first memory map; and

evaluating the number of likely cache misses ~~using a model of a direct-mapped cache~~ by passing the physical addresses for that one memory map to a model of a direct-mapped cache;

wherein the step of translating a program counter trace and evaluating the number of likely cache misses is repeated for each of the memory maps in a set to be evaluated.

6. (Previously Presented) The method of claim 5, wherein the memory map generation tool is also operable to generate a further set of memory maps for evaluation taking into account the results of evaluation of the first set of memory maps.

7. (Currently Amended) A memory map evaluation tool comprising:

a first component operable to generate a program counter trace from execution of a first version of a program according to a first memory map and to provide from that program

counter trace a converted format defining a memory location in association with a function and an offset within the function using the first memory map; and

a second component operable to translate the converted format of the program counter trace into physical addresses using one of the set of memory maps to be evaluated, different from the first memory map, and to evaluate the number of likely cache misses ~~using a model of a direct-mapped cache~~ by passing the physical addresses for that one memory map under evaluation to a model of a direct-mapped cache.

8. (Previously Presented) The tool of claim 7, configured in the form of program code means which, when executed on a computer, carry out the method steps of:

(a) executing a first version of the program according to a first memory map to generate a program counter trace;

(b) converting the program counter trace into a format defining a memory location in association with a function and an offset within the function using the first memory map;

(c) translating the program counter trace into physical addresses using one of the set of memory maps to be evaluated, different from the first memory map;

(d) evaluating the number of likely cache misses using a model of a direct-mapped cache for that one memory map; and

repeating steps (c) and (d) for each of the memory maps in the set.

9. (Canceled)

10. (Canceled)

11. (Currently Amended) ~~The method of claim 10,~~ A method of optimizing memory mapping for an instruction cache having an original memory map, comprising:

generating a plurality of alternate memory maps;

evaluating each of the plurality of alternate memory maps for the potential number of misses resulting from cache memory conflicts;

selecting at least one of the evaluated plurality of alternate memory maps and in accordance with predetermined criteria;

generating new alternate memory maps from the selected at least one of the evaluated plurality of alternate memory maps and the original map;

evaluating the new alternate memory maps; and

repeatedly generating and evaluating new alternate memory maps until an end criteria is met;

wherein the end criteria comprises one from among: a predetermined number of memory maps that have been evaluated, failure to find a better memory map, and when a ~~member~~ number of ~~missed instructions~~ misses is greater than a predetermined minimum number of ~~missed instructions~~ misses.

12. (Currently Amended) ~~The method of claim 10,~~ A method of optimizing memory mapping for an instruction cache having an original memory map, comprising:

generating a plurality of alternate memory maps;

evaluating each of the plurality of alternate memory maps for the potential number of misses resulting from cache memory conflicts;

selecting at least one of the evaluated plurality of alternate memory maps and in accordance with predetermined criteria;

generating new alternate memory maps from the selected at least one of the evaluated plurality of alternate memory maps and the original map;

evaluating the new alternate memory maps; and

repeatedly generating and evaluating new alternate memory maps until an end criteria is met;

wherein generating new alternate memory maps comprises one from among the following: swapping functions from the selected at least one of the plurality of alternate memory

maps and the original memory map, and selecting at least two of the alternate memory maps and merging the selected at least two of the alternate memory maps with the original memory map.

13. (Currently Amended) A software optimization method, comprising:

compiling a program;

generating a first memory map;

executing the program and generating a program count trace;

converting the program count trace to a program count trace format for finding a memory location in association with a function and an offset within the function using the first memory map; and

evaluating and selecting a memory map, further comprising:

translating the program counter trace format into physical addresses;

executing the translated trace on a cache model using the physical addresses;

~~determine~~ determining the number of cache misses; and

when the number of cache misses is acceptable, selecting the memory map and relinking to the program, otherwise selecting a new memory map and returning to the substep of translating the program count trace using the new memory map.

14. (Canceled)

15. (Currently Amended) ~~The method of claim 14,~~ A software optimization method, comprising:

selecting a plurality of memory maps for evaluation and end criteria for terminating the evaluation;

evaluating each of the memory maps until the end criteria for terminating the evaluation is met, further comprising:

evaluating the performance of each selected memory map; and

creating a new set of memory maps from the memory maps previously evaluated for a next evaluation;

wherein creating a new set of memory maps comprises selecting the memory map with the least number of ~~missed instructions~~ misses resulting from cache memory conflicts and performing a swap of two random functions therein.

16. (Currently Amended) ~~The method of claim 14,~~ A software optimization method, comprising:

selecting a plurality of memory maps for evaluation and end criteria for terminating the evaluation;

evaluating each of the memory maps until the end criteria for terminating the evaluation is met, further comprising:

evaluating the performance of each selected memory map; and

creating a new set of memory maps from the memory maps previously evaluated for a next evaluation;

wherein creating a new set of memory maps comprises selecting two or more memory maps having the best performance from the previous evaluation and merging the changes from the two selected memory maps to create a new memory map.

17. (Currently Amended) ~~The method of claim 14,~~ A software optimization method, comprising:

selecting a plurality of memory maps for evaluation and end criteria for terminating the evaluation;

evaluating each of the memory maps until the end criteria for terminating the evaluation is met, further comprising:

evaluating the performance of each selected memory map; and

creating a new set of memory maps from the memory maps previously evaluated for a next evaluation;

wherein the end criteria comprises one from among: a set number of evaluations that have been performed, a set number of misses that have been reached, and failure to find a better memory map after a predetermined number of evaluations.